Indexes and Constraints

/*Indexes

When you insert a row into a table, the database server does not attempt to put the data in any particular location within the table. For example, if you add a row to the customer table, the server doesn’t place the row in numeric order via the customer_id column or in alphabetical order via the last_name column. Instead, the server simply places the data in the next available location within the file. therefore, the server will need to inspect every row of the table to answer the query.*/

```
SELECT
    first_name, last_name
FROM
    customer
WHERE
    last_name LIKE 'y%';
```

/*In the same way that a person uses an index to find words within a book, a database server uses indexes to locate rows in a table. Indexes are special tables that, unlike normal data tables, are kept in a specific order.*/

/*Index Creation*/

```
ALTER TABLE customer
ADD INDEX index_email (email);
```

/*MySQL treats indexes as optional components of a table, which is why in earlier versions you would use the alter table command to add or remove an index.*/

```
SHOW INDEX FROM customer;
```

```
ALTER TABLE customer
DROP INDEX index_email;
```

/*Unique indexes

When designing a database, it is important to consider which columns are allowed to contain duplicate data and which are not. A unique index plays multiple roles; along with providing all the benefits of a regular index, it also serves as a mechanism for disallowing duplicate values in the indexed column.*/

```
ALTER TABLE customer
ADD UNIQUE idx_email (email);
```

```
INSERT INTO customer
(store_id, first_name, last_name, email, address_id, active)
VALUES
(1,'ALAN','KAHN', 'ALAN.KAHN@sakilacustomer.org', 394, 1);
```

/*You should not build unique indexes on your primary key column(s), since the server already checks uniqueness for primary key values.*/

/*Multicolumn indexes

Along with the single-column indexes demonstrated thus far, you may also build indexes that span multiple columns.*/

```
ALTER TABLE customer
ADD INDEX idx_full_name (last_name, first_name);
```

/*This index will be useful for queries that specify the first and last names or just the last name, but it would not be useful for queries that specify only the customer's first name.*/

/*Types of Indexes

B-tree indexes - balanced-tree indexes

B-tree indexes are organized as trees, with one or more levels of branch nodes leading to a single level of leaf nodes. Branch nodes are used for navigating the tree, while leaf nodes hold the actual values and location information.*/

/*Bitmap index (MySQL do not support this)

A bitmap index is a special kind of database index that uses bitmaps. Bitmap indexes have traditionally been considered to work well for low-cardinality columns (columns that contain only a small number of values across a large number of rows), which have a modest number of distinct values, either absolutely, or relative to the number of records that contain the data.*/

/*Text indexes

If your database stores documents, you may need to allow users to search for words or phrases in the documents. You certainly don't want the server to peruse each document and scan for the desired text each time a search is requested, but traditional indexing strategies don't work for this situation.*/

/*How Indexes are Used

Indexes are generally used by the server to quickly locate rows in a particular table, after which the server visits the associated table to extract the additional information requested by the user.*/

```
SELECT
    customer_id, first_name, last_name
FROM
    customer
WHERE
    first_name LIKE 'S%'
        AND last_name LIKE 'P%';
```

/*For this query, the server can employ any of the following strategies:

• Scan all rows in the customer table.

• Use the index on the last_name column to find all customers whose last name starts with P; then visit each row of the customer table to find only rows whose first name starts with S.

• Use the index on the last_name and first_name columns to find all customers whose last name starts with P and whose first name starts with S.*/

/*Constraints

A constraint is simply a restriction placed on one or more columns of a table.

Primary key constraints

Identify the column or columns that guarantee uniqueness within a table

Foreign key constraints

Restrict one or more columns to contain only values found in another table’s primary key column

Unique constraints

Restrict one or more columns to contain unique values within a table (primary key constraints are a special type of unique constraint)

Check constraints

Restrict the allowable values for a column*/

/*Exercise - 1

Generate an alter table statement for the rental table so that an error will be

raised if a row having a value found in the rental.customer_id column is deleted

from the customer table.

```
ALTER TABLE rental

ADD CONSTRAINT fk_rental_customer_id FOREIGN KEY (customer_id)

REFERENCES customer (customer_id) ON DELETE RESTRICT;*/
```

/*Exercise - 2

Generate a multicolumn index on the payment table that could be used by both of the

following queries:

```
SELECT customer_id, payment_date, amount

FROM payment

WHERE payment_date > cast('2019-12-31 23:59:59' as datetime);

SELECT customer_id, payment_date, amount

FROM payment

WHERE payment_date > cast('2019-12-31 23:59:59' as datetime)

AND amount < 5;

CREATE INDEX idx_payment01

ON payment (payment_date, amount);*/
```